International Journal on Emerging Technologies (Special Issue on ICRIET-2016) **7**(2): 288-292(2016)

ISSN No. (Print) : 0975-8364
ISSN No. (Online) : 2249-3255

Secure Message Transmission with the Implementation of RC4 & AES Cryptographic

Bhimrao Patil
Assistant Professor, Department of Computer Science Engineering,
BKIT Bhalki

(Corresponding author: Bhimrao Patil)
(Received 28 September, 2016 Accepted 29 October, 2016)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: Now a days Short Message Service (SMS) still represents the most used mobile messaging service. SMS messages are used in many different applications, even in cases where security features, such as authentication and confidentiality between the communicators must be ensured. Unfortunately, the SMS technology does not provide a built-in support for any security features. Though, it is claimed that all messages use a public key cryptography by the service provider, the messages are readable by the service provider. Hence secured communication over SMS is a big challenge. This work presents, a software framework written in .Net which allows two peers to exchange encrypted and digitally signed SMS messages. The communication between peers is secured by using public-key cryptography. The key-exchange process is implemented over a voice framework whereby the end users exchange the passkey over the phone. The passkey is than hashed using AES and the hashed password is used for encrypting the messages. For encryption we use RC4 and AES with Rijndael encryption engine. At the sender side, a Phone is configured as modem using GSM technique. The message is composed and encrypted and sent through GSM interface. At the receiver side, the message is firstly downloaded to PC through PC suite; decryption is than applied over the message. Special attention has been devoted to the implementation of an efficient framework in terms of energy consumption and execution time. This efficiency is obtained in two steps. First, all the cryptosystems available in the framework are implemented using mature and fully optimized cryptographic libraries. Second, an experimental analysis was conducted to determine which combination of cryptosystems and security parameters were able to provide a better trade-off in terms of speed/security and energy consumption.

I. INTRODUCTION

SMS messages are currently one of the most widespread forms communications (in 2008 about six trillion SMS were sent globally, see [1]). Sending an SMS is cheap, fast and simple. we have seen many unusual or strange applications, such as devices which allow the switching on and off of house heating systems using an SMS [2]. Alternatively, through SMS, whenever the temperature of a refrigerator exceeds a certain threshold, it is possible to automatically, communicate the problem [3], financial transactions (often, microtransactions) by sending SMS messages [4].Many of these services seem to ignore one important drawback of SMS based communication: the substantial lack of security. For example, by using bulk SMS service providers it is relatively simple to forge an SMS and send it to a recipient, as if it was transmitted by any sender.

So, services like the ones we mentioned before are prone to be attacked by malicious users [6]. In this case, for example, it would be easy to damage a user of the service by just sending to the Mobility Agency servers several forged SMS that have apparently been originated by that user. Two are the major security vulnerabilities affecting SMS based communication [7]: the lack of confidentiality during the transmission of a message and the absence of a standard way to certify the identity of the user (or at least his phone number) who sent the message. These vulnerabilities originate from the protocol used to exchange SMS messages and from the infrastructures used to implement it. There are currently several proposals, mostly coming from the scientific research, about how to secure SMS messages. Some of these proposals require security to be injected at the protocol level.

Instead, most of them consist of software frameworks which can be installed on mobile phones and/or on the SIM cards in order to implement security features. This paper presents a novel contribution to this field, consisting of a software framework which allows two peers (end users and/or software applications) to exchange SMS messages in a secure way, certifying the phone number of the peers involved in a SMS communication.

II. RELATED WORK

There have been several proposals up to now to secure SMS based communications on a GSM network. A first category of contributions tries to address these problems by changing the original GSM specifications in order to introduce security features. This is the case, for example, of the proposal presented by Hossain et al. in [7] which argues for a modification of the GSM protocol at the transport level to achieve confidentiality between mobile equipment (ME) and the GSM base station (BS) connected to it. The advantage of this approach, if followed, is that it would be able to inject security features at infra-structural level, thus allowing strengthening the entire communication network.

However, it is unlikely that these proposals will be implemented and widely adopted in the near future, mostly because of the technical difficulties arising from the implementation of structural changes in well established network architecture like the GSM one.

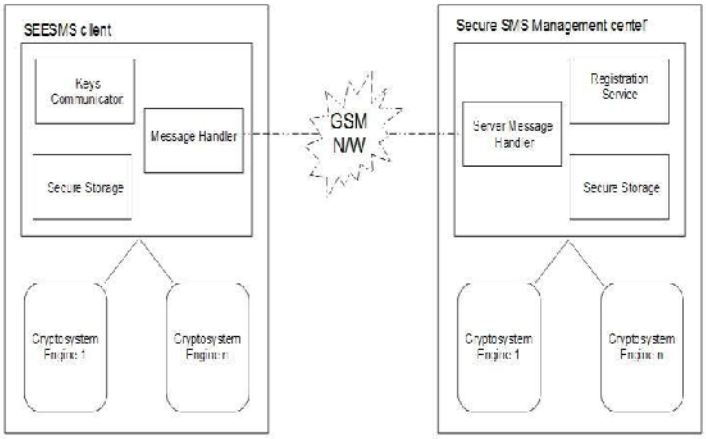
A second category of contributions to secure SMS communication- which is becoming viable because of the increasing diffusion of ME with advanced computational capabilities - introduces security features through the implementation of security schemes at the application level. The resulting software frameworks can be categorized according to the place where the application implementing the security scheme, and their cryptographic keys, are stored. The first possibility is to locate the application and its keys in a programmable SIM card used by the ME. This solution is adopted by systems like the one developed by Rongyu et al. in [8] or by the IPCS Group with the IPCryptSIM [9]. The use of a programmable SIM card has several advantages, such as the tamper resistance of the card and the possibility to move it from one ME to another without any data loss. However, it also has a relevant drawback: the limited computational capabilities of a programmable card do not allow the execution of complex security schemes within a reasonable amount of time.

An alternative approach, adopted in systems like the one presented in [10], is to use a SIM card only to store the cryptographic keys used in a scheme, while using the computational capabilities of the ME to run the scheme. In addition, it is also possible to use a SIM card to perform certain cryptographic operations, while executing the remaining part of the application through the ME, like in the mobile payment scheme presented by Hassinen et al. in [11]. Even just storing the cryptographic keys on a SIM card has an important disadvantage: the user is tied to the SIM provided by a particular operator and the intero-perations with SIM cards relative to other operators may be difficult or impossible to achieve.

SMSSec is an end-to-end protocol with the object of providing SMS security. It does not require any private-key to be stored in the mobile device, but provides user authentication and encryption by means of a PIN code running an ad hoc protocol with an Authentication Source (AS) authority. SMSSec uses symmetric (AES) and asymmetric (RSA) cryptography for the encryption and key-agreement respectively. On one hand, this approach allows a fast encryption process, while not altering the size of the SMS. Whereas, on the other, prior to any transactions, there is need of a new key-agreement with the AS, with the consequent exchange of additional initialization messages.

III. PROPOSED SYSTEM

The SEESMS framework adopts hybrid architecture. If a user is interested in sending/receiving a secure message through SEESMS and has never used it before, then he/she has to contact a trusted third-party server, called Secure SMS Management Center (SSMC), to request a customized copy of the SEESMS client application. Similarly, if the user has already installed the SEESMS client, but does not own the public-key of the recipient of the mssg (or the public-key of the user who sent him a secure SMS message), he/she has to contact the SSMC server to ask for a copy of his key (this behaviour is similar to the PGP key-servers). Instead, if the user already owns the public-key of his recipient, he/she will establish a direct communication in a peer-to-peer fashion, without further interaction with the SSMC server. Due to the use of a standard interface definition, all the cryptosystem engines have the same interface resulting in the ability to load them in the framework seamlessly. The following subsections describe in details of the SEESMS software components and their internal architecture, as described in above Figure 1 SECURE SMS MANAGEMENT CETRER: The SSMC is in charge of handling the provisioning pro-cess, used to deliver to new users a customized copy of the SEESMS client application, and the key-distribution process, used to send the public-keys of registered users following a client request. The entire communication with clients is done by using signed SMS messages.



The application includes the following modules:

Registration Service (RS): The RS is used to register new users, to provide them a copy of the SEESMS client application and to run key-exchange protocols with them.

Server Message Handler (SMH): The SMH is a module that can be used to exchange messages with another peer by means of SMS messages. It also includes the code needed to serialize /deserialize SMS messages and send /receive them through a GSM modem.

Secure Storage (SS): The SS implements a secure local storage area used to encrypt and to maintain sensitive data about the users that are registered to the service, such as their public-keys or their registration information. Data is encrypted using the AES symmetric cipher and stored in a relational database.

Cryptosystem Engines (CE): The CE are the modules that take care of securing the messages exchanged with a remote user. Each CE carries the implementation of a cryptosystem and offers up to three standard set of functions: Key Generation, Message Encryption/Decryption and Message Signature/Verification. These engines are used by the SSMC to implement the user registration phase and the key-exchange protocol. The current version of SEESMS includes the engines implementing ECC (ECDSA and ECIES), RSA and DSA. SEESMS CLIENT: The SEESMS client application can be used by two parties

To exchange encrypted and digitally signed SMS messages. It includes the following modules.

Message Handler (MH): The MH is responsible for sending and receiving secure SMS messages. It is a trimmed version of the SMH, not including the code needed to handle communication over a GSM modem.

Secure Storage (SS): The SS implements a secure local storage area used to hold sensitive data such as the cryptographic keys of a user.

Cryptosystem Engines: Similarly to the SSMC case, these modules are used to implement the registration phase and the key-exchange protocol and, moreover, all the functions related to secure communications with another user.

Keys Communicator: This module implements the client-side key-exchange protocol, which is used to communicate to the SSMC the cryptographic keys generated by the client.

RC4 Algorithm. RC4 is a stream cipher symmetric key algorithm. It was developed in 1987 by Ronald Rivest and kept as a trade secret by RSA Data Security. On September 9,1994, the RC4 algorithm was anonymously posted on the Internet on the Cyperpunks' "anonymous remailers" list. RC4 uses a variable length key from 1 to 256 bytes to initialize a 256-byte state table. The state table is used for subsequent generation of pseudo-random bytes and then to generate a pseudo-random stream which is XOR-ed with the plaintext to give the cipher text. Each element in the state table is swapped at least once. The RC4 key is often limited to 40 bits, because of export restrictions but it is sometimes used as a 128 bit key. It has the capability of using keys between 1 and 2048 bits. RC4 is used in many commercial software packages such as Lotus Notes and Oracle Secure SQL. It is also part of the Cellular Specification.

Algorithm Description: The RC4 algorithm works in two phases:

• Key setup
• ciphering

Key setup. Key setup is the first and most difficult phase of this algorithm. During a N-bit key setup (N being your key length), the encryption key is used to generate an encrypting variable using two arrays, state and key, and N-number of mixing operations. These mixing operations consist of swapping bytes, modulo operations, and other formulae.

The preliminary operations can be summarized as follows:

```
//Initialization for
i=0 to 255 do s[i]=i;
T[i]=K[i mod keylen];
```

Next use T to produce the initial permutation of S. this involves starting with s[0] and going through s[255], and for each of s[i], swapping S[i] with another byte in S according to scheme dictated by t[i]:

```
// Initial Permutation of s
j=0;
for i=0 to 255 do
j=(j+S[i]+T[i]) mod 256;
Swap(S[i],s[j]);
//stream generation
i,j=0;
while(true)
i= (i+1) mod 255;
j=(j+S[i]) mod 256;
Swap(S[i],S[j]);
t=(s[i]+S[j]) mod 256;
k=([t]
```

In the attached project you can see how I do it in the Encryption Key set property of RC4Engine class.

Ciphering Phase. Once the encrypting variable is produced from the key setup, it enters the ciphering phase, where it is XOR-ed with the plain text message to create an encrypted message. XOR is the logical operation of comparing two binary bits. If the bits are different, the result is 1. If the bits are the same, the result is 0. Once the receiver gets the encrypted message, he decrypts it by XOR-ing the encrypted message with the same encrypting variable.

In the attached project you can see how I do it in the RC4Engine class:

• Encrypt: encript method

• Decrypt: decript method

AES. AES is short for Advanced Encryption Standard.AES is a symmetric encryption algorithm that us certain substitution & permutation methods to process data in block of 128 bits.AES may configured to use different key-lengths, the standard defines 3 lengths and the resulting algorithms are named AES-128, AES-192 and AES-256 respectively to indicate the length in bits of the key. Each additional bit in the key effectively doubles the strength of the algorithm, when defined as the time necessary for an attacker to stage a brute force attack**.**

Cryptographic Algorithms.

Algorithm Name	Structure	Key Size (In bits)	Rounds Cipher	Type
AES Substitution	permutation network	128, 192, 256	10, 12, 14	Block
RC4		40 to 2048	256	Stream

IV. CONCLUSIONS

There are several proposals for securing the short messages due to inherent unsecured framework of the same. As SMS is considered to be for exchanging shorter information, the core protocol does not adopt any security. As the digital age has evolved, SMS starts supporting larger messages (more than 160 characters) by fragmenting the one message into smaller message and then joining them together at the receiver mobile. Hence more critical data are being exchanged over SMS like bank account details, passwords of the accounts, loan number and so on. This makes the system vulnerable as the intruder can pick a gateway to hack the messages and extract information. But the conventional solutions offer the technique to be implemented in the mobile using technologies like J2ME or Android. But the results clearly shows that the encryption and decryption consumes a lot of energy which is far more significant if carried out at the mobiles due to the hardware limitations of the mobile.

Therefore in this work I have demonstrated a unique mechanism for SMS security through adopting RC4 and Rijndael based security. The result analysis clearly shows that RC4 is better in terms of energy conservation whereas Rijndael provides better encryption strength. currently mobile equipment(ME) alone will not able to decrypt cipher text at destination (receiver side).So in future ME can be enhanced such that it can be able to decrypt the message on its own , without using the PC suite software and an extra hardware like Personal computer.

This work advises that before implementing a cryptographic application by using existing cryptographic libraries, it would be a good practice to test if the experimental performances of these libraries follow the expected theoretical results.

REFERENCES

[1]. Mobile Message Analyst, http://www.bizcommunity.com/Article/196/78/38788.html, online visited July 2009.

[21]. Tele-Log, http://www.tele- log.com/domotica-e.html , online visited July 2009.

[3]. A. Castiglione, R. De Prisco, and A. De Santis, ”Do You Trust Your Phone?” in EC-Web 2009: Proceedings of the 10th International Conference on E-Commerce and Web Technologies, vol. LNCS 5692. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 50-61.

[4]. A. Hossain, S. Jahan, M. Hussain, M. Amin, and S. Shah Newaz, ”A proposal for enhancing the security system of short message service in GSM,” in Anticounterfeiting, Security and Identification, 2008. ASID 2008. 2nd International Conference on, Aug. 2008, pp. 235-240.

[5]. IPCS Group”IPCryptSim SMS Encryption”, http://www.ipcslive.com/pdf/IPCSSMS.pdf, online visited July 2009.

[6]. M. Toorani and A. Beheshti Shirazi, ”SSMS - A secure SMS messaging protocol for the m-payment systems,” in Computers and Communications, 2008. ISCC 2008. IEEE Symposium on, July 2008, pp.700

[7]. M. Hassinen, K. Hypponen, and K. Haataja, An Open, PKI-Based Mobile Payment System, in ETRICS, 2006, pp. 86100.

[8]. M. Hassinen, SafeSMS - end-to-end encryption for SMS, in Telecommunications, 2005. ConTEL 2005. Proceedings of the 8th International Conference on, vol. **2**, 15-17, 2005, pp. 359365.

[9]. D. Lisonek and M. Drahansky, SMS Encryption for Mobile Communication, in Security Technology, 2008. SECTECH 08. International Conference on, Dec. 2008, pp. 19Polymorphic virus8201.